MOTOROLA
Semiconductor Products Inc.

AN-819
Application Note

PRIORITIZED INDIVIDUALLY VECTORED INTERRUPTS FOR MULTIPLE PERIPHERAL SYSTEMS WITH THE MC68000

Prepared by:
Rex Davis
Microprocessor Applications Engineer
Austin, Texas

Commercial and industrial microprocessor systems typically consists of a processor interfaced with some type of peripherals which, most of the time, require service from the processor. When a peripheral requires service, it flags the processor with an interrupt request. The processor will determine if the interrupt is to be serviced by checking an interrupt mask.

If the mask is not set, then the processor has an interrupt service timing requirement which, along with the data rate and interrupt frequency, can be used to determine the relative priority of each of the peripherals in the system. If two or more peripherals attempt to request service simultaneously, the relative priority of each peripheral determines which peripheral receives service first. In order to minimize the risk of violating timing requirements of lower priority peripherals, the processor must quickly identify and service the current highest priority interrupt. The address of the service routine is contained in a vector; and every interrupt source should have a unique vector for its service routine.

MC68000 INTERRUPT STRUCTURE

Interrupt requests are input to the MC68000 through three pins which represent seven levels of interrupt priority and a quiescent state (no interrupt). The MC68000 status register contains a three-bit mask which only enables interrupts of a higher priority than the level represented in the three-bit mask. The interrupt mask can be modified under software control, thereby increasing user control of peripheral interrupt requests. The user's hardware generates an interrupt request by encoding the peripheral interrupt into one of seven interrupt priority levels and driving the interrupt request lines with the three-bit representation of that priority level.

Figure 1 is a flow chart of the MC68000 interrupt process. Interrupt requests are considered by the MC68000 to be pending until the completion of the current instruction execution. If, at that time, the priority of the pending interrupt is less than or equal to the current processor priority represented by the three-bit interrupt mask, then the next instruction is executed. If the priority of the pending interrupt is greater than the processor priority, then interrupt exception processing begins. During interrupt exception processing, the MC68000 places the level of interrupt priority on address lines A1, A2, and A3. These lines can be used to quickly determine which group of peripherals might have generated the interrupt request. Simultaneously, the function code outputs (FC0-FC2) are set to indicate an interrupt acknowledge (IACK) which flags the user hardware that exception processing has begun. The MC68000 architecture uses the first 1024 bytes of memory for vector storage.

Any exception to free-running operation, such as an interrupt, has a vector stored at a unique location in the 1024 byte exception memory map. Exceptions other than interrupts, include reset, system errors, software traps, and unimplemented instruction emulators, as shown in Table 1. Since each vector, except reset, requires a 32-bit address, four bytes are required to store each vector. Reset is a special case which requires two 32-bit addresses or eight bytes of memory.

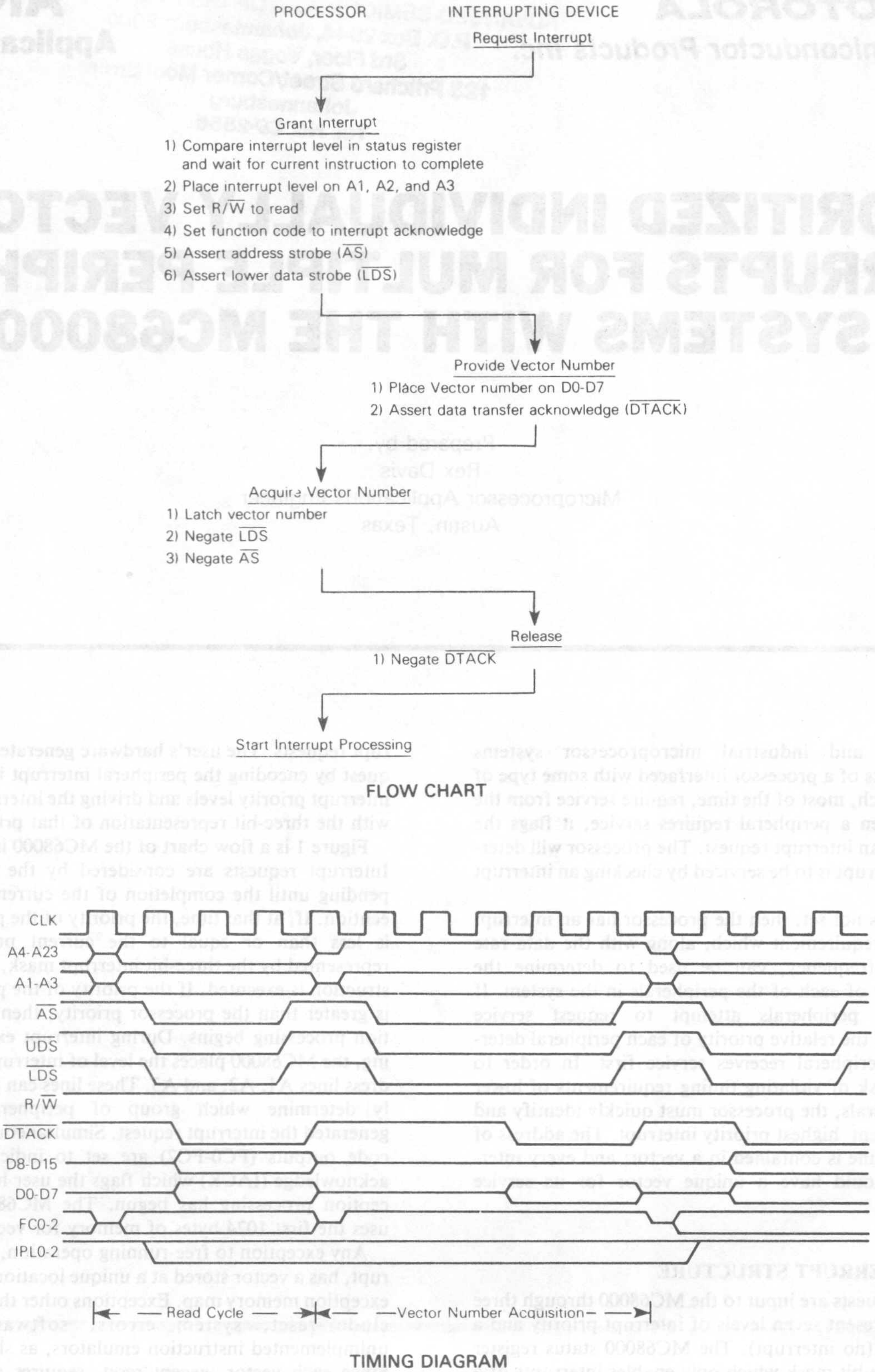

Figure 1. MC68000 Interrupt Acknowledge Sequence — Flow Chart and Timing Diagram

Vector Number(s)	Address Dec	Address Hex	Space	Assignment
0	0	000	SP	Reset: Initial SSP
1	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		—
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

*Vector numbers 12 through 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

Table 1. Exception Vector Assignments

The exception vector map can be divided into 255 unique vectors which can be represented by an eight-bit vector number. The vector number is not the vector; it is a pointer to one particular vector. During any exception processing, the MC68000 fetches the vector pointed to by a vector number. Each exception has a unique vector number which, for all exceptions except user interrupts is generated internally by the MC68000. User interrupts require that the vector number be placed on the lower eight bits of the data bus during interrupt acknowledge. The addition of the vector number fetch requires the peripheral to supply only the eight-bit vector number instead of the whole 32-bit vector.

The MC68000 allows two methods of interrupt vector number generation, internal or external. In order to generate the vector number internally, $\overline{\text{VPA}}$ is connected to IACK. In this mode, a unique vector number is generated for each interrupt priority level. This mode, called the autovector mode, is ideal for users requiring less than eight levels of interrupts or users with more than seven peripherals whose timing requirements are non-critical.

For users with more than seven peripherals and whose timing requirements demand service faster than possible with a software polling method, the MC68000 provides an additional 192 interrupts which require external vector number generation. In this case, IACK is not connected to $\overline{\text{VPA}}$; instead, it is used by external hardware to determine that a vector number is needed by the MC68000 and to provide the proper vector number and data transfer acknowledge ($\overline{\text{DTACK}}$).

In systems with only slightly more than seven possible interrupt sources, IACK is first sent to the highest priority peripheral and then daisy-chained to all remaining peripherals in order of priority. If any peripheral generates an interrupt, it must suppress IACK to all remaining peripherals and then place the vector number on the lower eight bits of the MC68000 data bus and assert $\overline{\text{DTACK}}$. Systems that have a large number of interrupt sources and timing requirements too critical for software polling can also generate the vector number and $\overline{\text{DTACK}}$ by the same method. However, since every peripheral must have the capability to generate these signals, redundant hardware is spread throughout the system making debug, modification, and maintenance difficult. Alternatively, all interrupt lines could be brought to a central location where both $\overline{\text{DTACK}}$ and the proper vector number could be supplied. The application describes a system that will provide $\overline{\text{DTACK}}$ and the vector number for up to 192 possible interrupt sources.

VECTOR NUMBER GENERATION

The 192 user interrupt vectors are referenced by sequential vector numbers 64 through 255. Therefore, if fewer than 193 interrupts are required, each interrupt can be assigned a unique vector number which can also be interpreted as a priority. Vector number 64 can be assumed to have the

lowest priority and vector number 255 the highest. The 192 levels of externally-generated priority can be represented by eight bits (00000000-10111111). The vector number is the externally-generated priority offset by 64; the vector number can be generated by encoding the interrupt to its priority and then adding 64. This is essentially the same format that is used in the autovectors.

Figure 2 is a block diagram of such a system. All circuitry, except the processor, can be located in one area rather than spread throughout the system. Note that two sets of latches have been inserted to guarantee that no interrupts are lost and that the vector number that is placed on the data bus is the result of only one interrupt. Otherwise, if an interrupt request is generated during interrupt acknowledge, it could cause the vector number to be in a state of transition when the MC68000 is attempting to latch the vector number from the data bus. Latch number one prohibits any new interrupts from being accepted until the vector number has been latched by latch number two. Latch number two isolates the vector number from the data bus until IACK is asserted. After a delay sufficient to allow the vector number to propagate to latch number two, latch number one is released to allow new interrupts to be accepted.

IMPLEMENTATION

The circuitry shown in Figure 3 performs all the tasks necessary to provide vector numbers for up to 192 possible interrupt sources. The 192 interrupt request lines are divided into 24 groups of eight which are input through a SN74LS373 octal latch to a SN74LS148 8-to-3 encoder. The SN74LS148 encoders are daisy-chained so that each stage can disable all succeeding stages which effectively prioritizes the interrupts by group. Within each group, interrupts are prioritized into eight levels which the encoder represents with a three-bit encoded number on lines A0, A1, and A2.

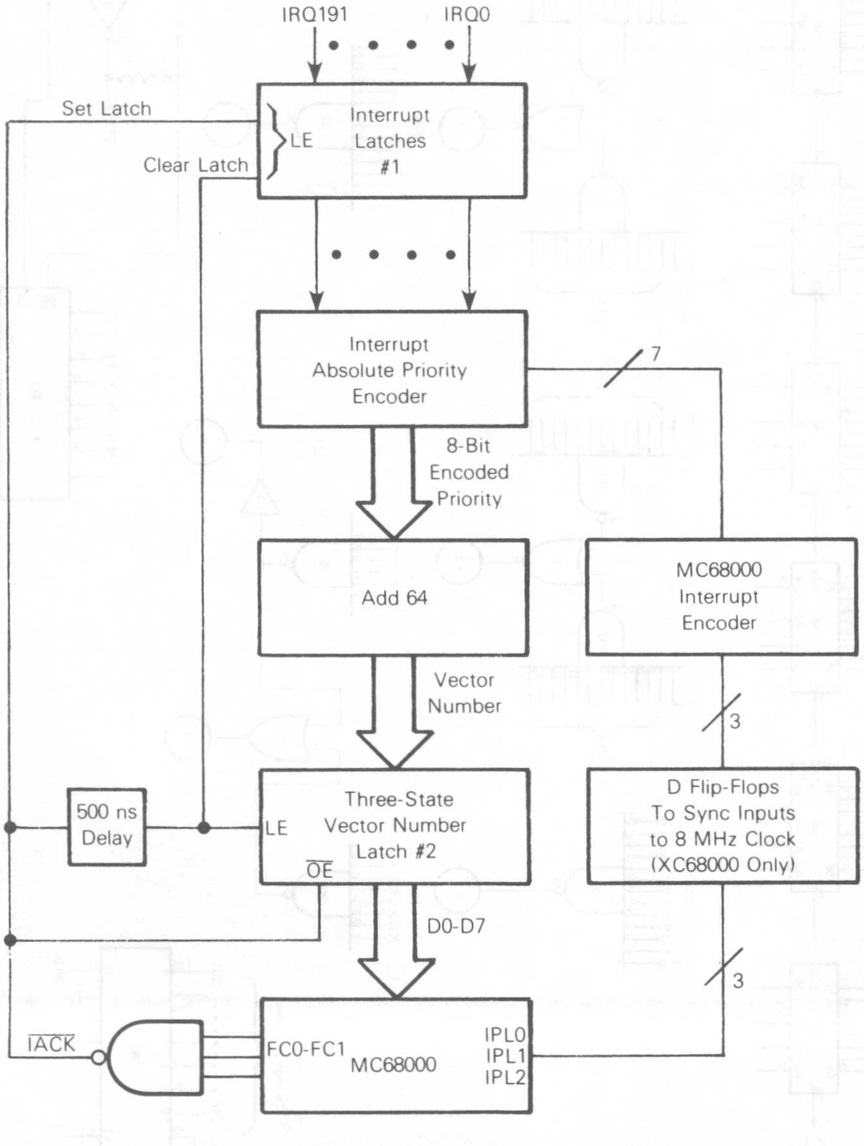

Figure 2. Vector Number Generation Circuit — Block Diagram

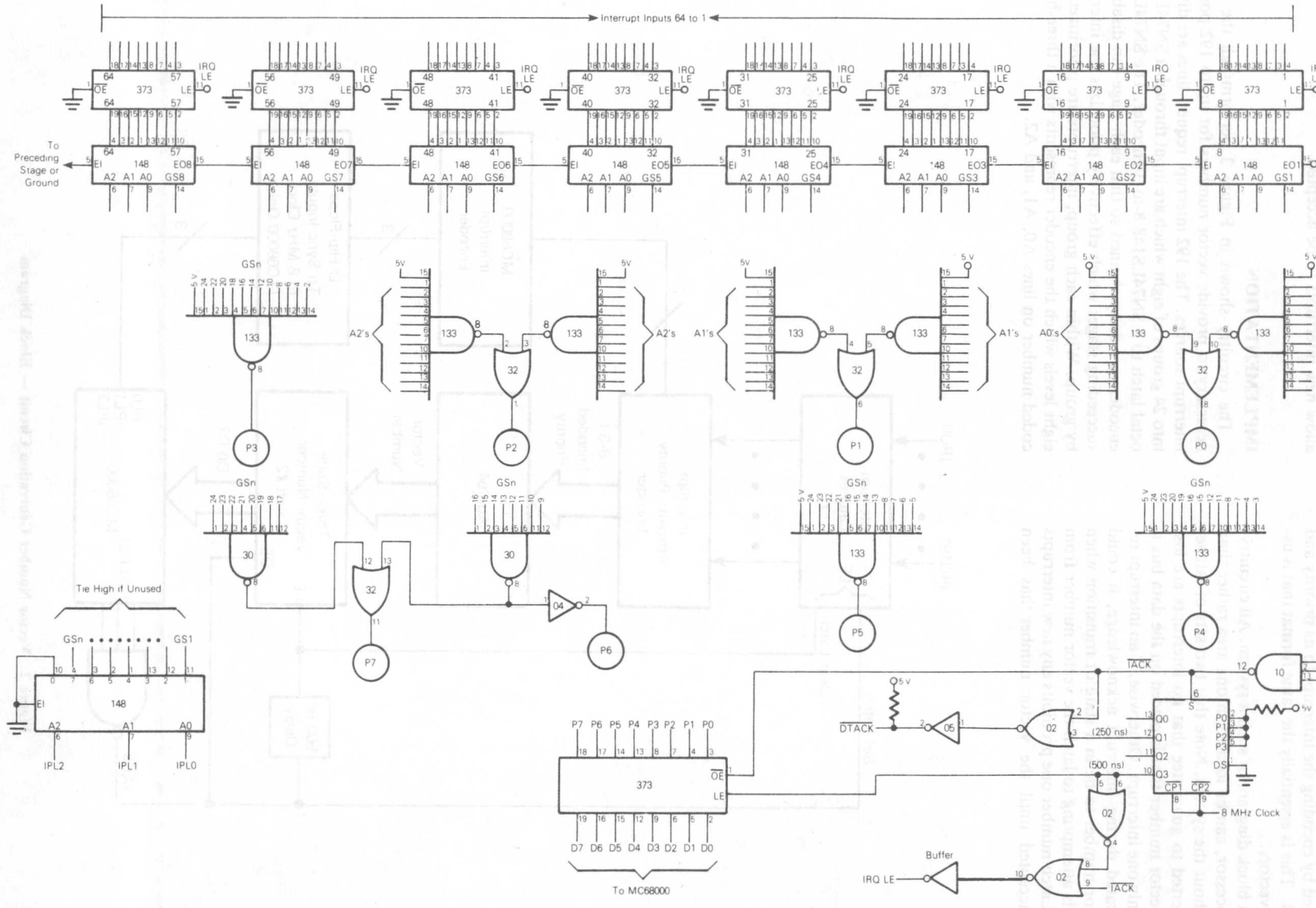

Figure 3. Vector Number Generation Circuit — Schematic Diagram

The A0 line from each of the 24 groups is NANDed to form the A0 of the vector number. The A1 and A2 lines are handled in an identical manner. Bits 3 and 7 of the encoded interrupt are made by NANDing selected GS outputs of the encoders. Bits 6 and 7 of the vector number differ from bits 6 and 7 of the encoded interrupt because of the offset of 64.

After the vector number has had sufficient time to propagate, a second SN74LS373 octal latch is used to capture the number and allow the latches in the 24 interrupt group to be released to accept new interrupt requests. The delay, imposed by a SN74LS95 four-bit shift register to latch the vector number, assumes that all 24 groups are implemented and the MC68000 is running at eight megahertz. Timing requirements can be derived from Figure 4.

A final SN74LS148 is used to encode the three-bit interrupt requests to the MC68000. The inputs to this encoder are the GS outputs of the SN74LS148 encoders from any group of interrupts. The group providing the GS output and all preceding groups can activate the level to which the GS output is input. However, GS outputs of preceding groups which are input to a higher level in the final SN74LS148 will effectively disable the lower level interrupt request.

If R9M or T6E mask types are used in the system, then the seven levels of interrupts must be latched before encoding on the rising edge of the processor clock. The latch is necessary to synchronize the interrupt request lines.

VARIATIONS

The following variations to the system given in the application can be considered in light of specific system requirements:

1. Flip-flops could be inserted on each group to latch an edge-type interrupt input.
2. The level of interrupt that initiated exception processing could be decoded from address lines A1, A2, and A3 of the MC68000.
3. If vector numbers reserved for other functions, e.g., TRAPS, auto vectors, are unused; then they could be used for user interrupts. However, observe caution when using any reserved vector numbers.

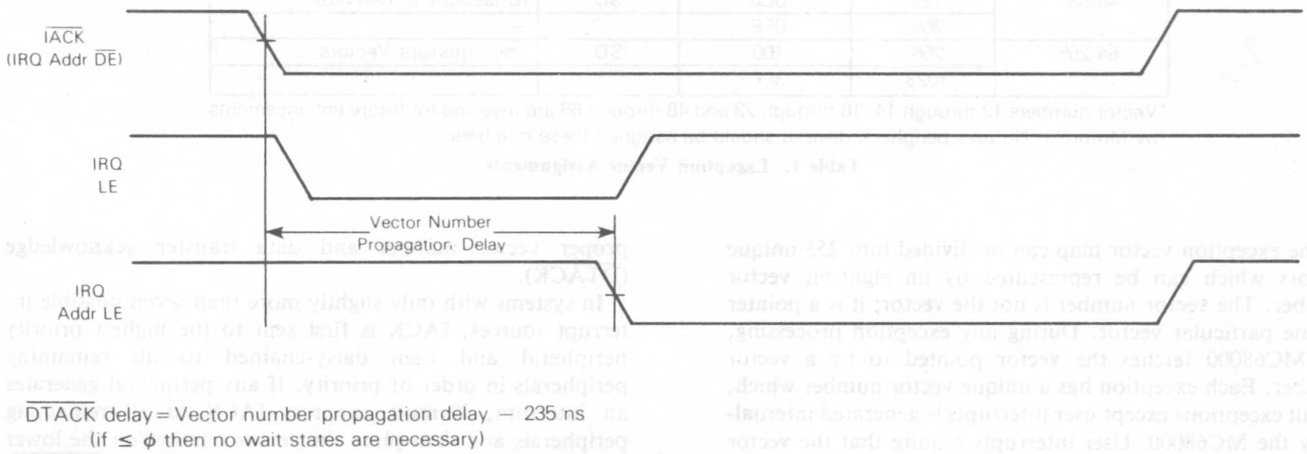

Figure 4. Vector Number Generation Circuit — Timing Diagrams